EXHIBIT B20

```
/*
 * Copyright (c) 2015-2016, Freescale Semiconductor, Inc.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
 *
 * o Redistributions of source code must retain the above copyright notice, this list
 *   of conditions and the following disclaimer.
 *
 * o Redistributions in binary form must reproduce the above copyright notice, this
 *   list of conditions and the following disclaimer in the documentation and/or
 *   other materials provided with the distribution.
 *
 * o Neither the name of Freescale Semiconductor, Inc. nor the names of its
 *   contributors may be used to endorse or promote products derived from this
 *   software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
 * ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
```

```
30
31  #ifndef __CCM_IMX7D_H__
32  #define __CCM_IMX7D_H__
33
34  #include <stdint.h>
35  #include <stdbool.h>
36  #include <stddef.h>
37  #include <assert.h>
38  #include "device_imx.h"
39
40  /*!
41   * @addtogroup ccm_driver
42   * @{
43   */
44
45  /*******************************************************************************
46   * Definitions
47   ******************************************************************************/
48  #define CCM_REG_OFF(root, off) (*((volatile uint32_t *)((uint32_t)root + off)))
49  #define CCM_REG(root)          CCM_REG_OFF(root, 0)
50  #define CCM_REG_SET(root)      CCM_REG_OFF(root, 4)
51  #define CCM_REG_CLR(root)      CCM_REG_OFF(root, 8)
52
53  /*! @brief Root control names for root clock setting. */
54  enum _ccm_root_control
55  {
56      ccmRootM4     = (uint32_t)(&CCM_TARGET_ROOT1),   /*!< ARM Cortex-M4 Clock control name.*/
57      ccmRootAxi    = (uint32_t)(&CCM_TARGET_ROOT16),  /*!< AXI Clock control name.*/
58      ccmRootAhb    = (uint32_t)(&CCM_TARGET_ROOT32),  /*!< AHB Clock control name.*/
59      ccmRootIpg    = (uint32_t)(&CCM_TARGET_ROOT33),  /*!< IPG Clock control name.*/
60      ccmRootQspi   = (uint32_t)(&CCM_TARGET_ROOT85),  /*!< QSPI Clock control name.*/
61      ccmRootCan1   = (uint32_t)(&CCM_TARGET_ROOT89),  /*!< CAN1 Clock control name.*/
62      ccmRootCan2   = (uint32_t)(&CCM_TARGET_ROOT90),  /*!< CAN2 Clock control name.*/
63      ccmRootI2c1   = (uint32_t)(&CCM_TARGET_ROOT91),  /*!< I2C1 Clock control name.*/
64      ccmRootI2c2   = (uint32_t)(&CCM_TARGET_ROOT92),  /*!< I2C2 Clock control name.*/
65      ccmRootI2c3   = (uint32_t)(&CCM_TARGET_ROOT93),  /*!< I2C3 Clock control name.*/
66      ccmRootI2c4   = (uint32_t)(&CCM_TARGET_ROOT94),  /*!< I2C4 Clock control name.*/
67      ccmRootUart1  = (uint32_t)(&CCM_TARGET_ROOT95),  /*!< UART1 Clock control name.*/
68      ccmRootUart2  = (uint32_t)(&CCM_TARGET_ROOT96),  /*!< UART2 Clock control name.*/
69      ccmRootUart3  = (uint32_t)(&CCM_TARGET_ROOT97),  /*!< UART3 Clock control name.*/
70      ccmRootUart4  = (uint32_t)(&CCM_TARGET_ROOT98),  /*!< UART4 Clock control name.*/
71      ccmRootUart5  = (uint32_t)(&CCM_TARGET_ROOT99),  /*!< UART5 Clock control name.*/
72      ccmRootUart6  = (uint32_t)(&CCM_TARGET_ROOT100), /*!< UART6 Clock control name.*/
73      ccmRootUart7  = (uint32_t)(&CCM_TARGET_ROOT101), /*!< UART7 Clock control name.*/
74      ccmRootEcspi1 = (uint32_t)(&CCM_TARGET_ROOT102), /*!< ECSPI1 Clock control name.*/
75      ccmRootEcspi2 = (uint32_t)(&CCM_TARGET_ROOT103), /*!< ECSPI2 Clock control name.*/
76      ccmRootEcspi3 = (uint32_t)(&CCM_TARGET_ROOT104), /*!< ECSPI3 Clock control name.*/
77      ccmRootEcspi4 = (uint32_t)(&CCM_TARGET_ROOT105), /*!< ECSPI4 Clock control name.*/
78      ccmRootFtm1   = (uint32_t)(&CCM_TARGET_ROOT110), /*!< FTM1 Clock control name.*/
```

```
    ccmRootFtm2   = (uint32_t)(&CCM_TARGET_ROOT111), /*!< FTM2 Clock control name.*/
    ccmRootGpt1   = (uint32_t)(&CCM_TARGET_ROOT114), /*!< GPT1 Clock control name.*/
    ccmRootGpt2   = (uint32_t)(&CCM_TARGET_ROOT115), /*!< GPT2 Clock control name.*/
    ccmRootGpt3   = (uint32_t)(&CCM_TARGET_ROOT116), /*!< GPT3 Clock control name.*/
    ccmRootGpt4   = (uint32_t)(&CCM_TARGET_ROOT117), /*!< GPT4 Clock control name.*/
    ccmRootWdog   = (uint32_t)(&CCM_TARGET_ROOT119), /*!< WDOG Clock control name.*/
};

/*! @brief Clock source enumeration for ARM Cortex-M4 core. */
enum _ccm_rootmux_m4
{
    ccmRootmuxM4Osc24m      = 0U, /*!< ARM Cortex-M4 Clock from OSC 24M.*/
    ccmRootmuxM4SysPllDiv2  = 1U, /*!< ARM Cortex-M4 Clock from SYSTEM PLL divided by 2.*/
    ccmRootmuxM4EnetPll250m = 2U, /*!< ARM Cortex-M4 Clock from Ethernet PLL 250M.*/
    ccmRootmuxM4SysPllPfd2  = 3U, /*!< ARM Cortex-M4 Clock from SYSTEM PLL PFD2.*/
    ccmRootmuxM4DdrPllDiv2  = 4U, /*!< ARM Cortex-M4 Clock from DDR PLL divided by 2.*/
    ccmRootmuxM4AudioPll    = 5U, /*!< ARM Cortex-M4 Clock from AUDIO PLL.*/
    ccmRootmuxM4VideoPll    = 6U, /*!< ARM Cortex-M4 Clock from VIDEO PLL.*/
    ccmRootmuxM4UsbPll      = 7U, /*!< ARM Cortex-M4 Clock from USB PLL.*/
};

/*! @brief Clock source enumeration for AXI bus. */
enum _ccm_rootmux_axi
{
    ccmRootmuxAxiOsc24m      = 0U, /*!< AXI Clock from OSC 24M.*/
    ccmRootmuxAxiSysPllPfd1  = 1U, /*!< AXI Clock from SYSTEM PLL PFD1.*/
    ccmRootmuxAxiDdrPllDiv2  = 2U, /*!< AXI Clock DDR PLL divided by 2.*/
    ccmRootmuxAxiEnetPll250m = 3U, /*!< AXI Clock Ethernet PLL 250M.*/
    ccmRootmuxAxiSysPllPfd5  = 4U, /*!< AXI Clock SYSTEM PLL PFD5.*/
    ccmRootmuxAxiAudioPll    = 5U, /*!< AXI Clock AUDIO PLL.*/
    ccmRootmuxAxiVideoPll    = 6U, /*!< AXI Clock VIDEO PLL.*/
    ccmRootmuxAxiSysPllPfd7  = 7U, /*!< AXI Clock SYSTEM PLL PFD7.*/
};

/*! @brief Clock source enumeration for AHB bus. */
enum _ccm_rootmux_ahb
{
    ccmRootmuxAhbOsc24m      = 0U, /*!< AHB Clock from OSC 24M.*/
    ccmRootmuxAhbSysPllPfd2  = 1U, /*!< AHB Clock from SYSTEM PLL PFD2.*/
    ccmRootmuxAhbDdrPllDiv2  = 2U, /*!< AHB Clock from DDR PLL divided by 2.*/
    ccmRootmuxAhbSysPllPfd0  = 3U, /*!< AHB Clock from SYSTEM PLL PFD0.*/
    ccmRootmuxAhbEnetPll125m = 4U, /*!< AHB Clock from Ethernet PLL 125M.*/
    ccmRootmuxAhbUsbPll      = 5U, /*!< AHB Clock from USB PLL.*/
    ccmRootmuxAhbAudioPll    = 6U, /*!< AHB Clock from AUDIO PLL.*/
    ccmRootmuxAhbVideoPll    = 7U, /*!< AHB Clock from VIDEO PLL.*/
};

/*! @brief Clock source enumeration for IPG bus. */
enum _ccm_rootmux_ipg
```

```
{
    ccmRootmuxIpgAHB = 0U, /*!< IPG Clock from AHB Clock.*/
};

/*! @brief Clock source enumeration for QSPI peripheral. */
enum _ccm_rootmux_qspi
{
    ccmRootmuxQspiOsc24m      = 0U, /*!< QSPI Clock from OSC 24M.*/
    ccmRootmuxQspiSysPllPfd4  = 1U, /*!< QSPI Clock from SYSTEM PLL PFD4.*/
    ccmRootmuxQspiDdrPllDiv2  = 2U, /*!< QSPI Clock from DDR PLL divided by 2.*/
    ccmRootmuxQspiEnetPll500m = 3U, /*!< QSPI Clock from Ethernet PLL 500M.*/
    ccmRootmuxQspiSysPllPfd3  = 4U, /*!< QSPI Clock from SYSTEM PLL PFD3.*/
    ccmRootmuxQspiSysPllPfd2  = 5U, /*!< QSPI Clock from SYSTEM PLL PFD2.*/
    ccmRootmuxQspiSysPllPfd6  = 6U, /*!< QSPI Clock from SYSTEM PLL PFD6.*/
    ccmRootmuxQspiSysPllPfd7  = 7U, /*!< QSPI Clock from SYSTEM PLL PFD7.*/
};

/*! @brief Clock source enumeration for CAN peripheral. */
enum _ccm_rootmux_can
{
    ccmRootmuxCanOsc24m     = 0U, /*!< CAN Clock from OSC 24M.*/
    ccmRootmuxCanSysPllDiv4 = 1U, /*!< CAN Clock from SYSTEM PLL divided by 4.*/
    ccmRootmuxCanDdrPllDiv2 = 2U, /*!< CAN Clock from SYSTEM PLL divided by 2.*/
    ccmRootmuxCanSysPllDiv1 = 3U, /*!< CAN Clock from SYSTEM PLL divided by 1.*/
    ccmRootmuxCanEnetPll40m = 4U, /*!< CAN Clock from Ethernet PLL 40M.*/
    ccmRootmuxCanUsbPll     = 5U, /*!< CAN Clock from USB PLL.*/
    ccmRootmuxCanExtClk1    = 6U, /*!< CAN Clock from External Clock1.*/
    ccmRootmuxCanExtClk34   = 7U, /*!< CAN Clock from External Clock34.*/
};

/*! @brief Clock source enumeration for ECSPI peripheral. */
enum _ccm_rootmux_ecspi
{
    ccmRootmuxEcspiOsc24m      = 0U, /*!< ECSPI Clock from OSC 24M.*/
    ccmRootmuxEcspiSysPllDiv2  = 1U, /*!< ECSPI Clock from SYSTEM PLL divided by 2.*/
    ccmRootmuxEcspiEnetPll40m  = 2U, /*!< ECSPI Clock from Ethernet PLL 40M.*/
    ccmRootmuxEcspiSysPllDiv4  = 3U, /*!< ECSPI Clock from SYSTEM PLL divided by 4.*/
    ccmRootmuxEcspiSysPllDiv1  = 4U, /*!< ECSPI Clock from SYSTEM PLL divided by 1.*/
    ccmRootmuxEcspiSysPllPfd4  = 5U, /*!< ECSPI Clock from SYSTEM PLL PFD4.*/
    ccmRootmuxEcspiEnetPll250m = 6U, /*!< ECSPI Clock from Ethernet PLL 250M.*/
    ccmRootmuxEcspiUsbPll      = 7U, /*!< ECSPI Clock from USB PLL.*/
};

/*! @brief Clock source enumeration for I2C peripheral. */
enum _ccm_rootmux_i2c
{
    ccmRootmuxI2cOsc24m         = 0U, /*!< I2C Clock from OSC 24M.*/
    ccmRootmuxI2cSysPllDiv4     = 1U, /*!< I2C Clock from SYSTEM PLL divided by 4.*/
    ccmRootmuxI2cEnetPll50m     = 2U, /*!< I2C Clock from Ethernet PLL 50M.*/
```

```
    ccmRootmuxI2cDdrPllDiv2     = 3U, /*!< I2C Clock from DDR PLL divided by .*/
    ccmRootmuxI2cAudioPll       = 4U, /*!< I2C Clock from AUDIO PLL.*/
    ccmRootmuxI2cVideoPll       = 5U, /*!< I2C Clock from VIDEO PLL.*/
    ccmRootmuxI2cUsbPll         = 6U, /*!< I2C Clock from USB PLL.*/
    ccmRootmuxI2cSysPllPfd2Div2 = 7U, /*!< I2C Clock from SYSTEM PLL PFD2 divided by 2.*/
};

/*! @brief Clock source enumeration for UART peripheral. */
enum _ccm_rootmux_uart
{
    ccmRootmuxUartOsc24m      = 0U, /*!< UART Clock from OSC 24M.*/
    ccmRootmuxUartSysPllDiv2  = 1U, /*!< UART Clock from SYSTEM PLL divided by 2.*/
    ccmRootmuxUartEnetPll40m  = 2U, /*!< UART Clock from Ethernet PLL 40M.*/
    ccmRootmuxUartEnetPll100m = 3U, /*!< UART Clock from Ethernet PLL 100M.*/
    ccmRootmuxUartSysPllDiv1  = 4U, /*!< UART Clock from SYSTEM PLL divided by 1.*/
    ccmRootmuxUartExtClk2     = 5U, /*!< UART Clock from External Clock 2.*/
    ccmRootmuxUartExtClk34    = 6U, /*!< UART Clock from External Clock 34.*/
    ccmRootmuxUartUsbPll      = 7U, /*!< UART Clock from USB PLL.*/
};

/*! @brief Clock source enumeration for FlexTimer peripheral. */
enum _ccm_rootmux_ftm
{
    ccmRootmuxFtmOsc24m      = 0U, /*!< FTM Clock from OSC 24M.*/
    ccmRootmuxFtmEnetPll100m = 1U, /*!< FTM Clock from Ethernet PLL 100M.*/
    ccmRootmuxFtmSysPllDiv4  = 2U, /*!< FTM Clock from SYSTEM PLL divided by 4.*/
    ccmRootmuxFtmEnetPll40m  = 3U, /*!< FTM Clock from Ethernet PLL 40M.*/
    ccmRootmuxFtmAudioPll    = 4U, /*!< FTM Clock from AUDIO PLL.*/
    ccmRootmuxFtmExtClk3     = 5U, /*!< FTM Clock from External Clock 3.*/
    ccmRootmuxFtmRef1m       = 6U, /*!< FTM Clock from Refernece Clock 1M.*/
    ccmRootmuxFtmVideoPll    = 7U, /*!< FTM Clock from VIDEO PLL.*/
};

/*! @brief Clock source enumeration for GPT peripheral. */
enum _ccm_rootmux_gpt
{
    ccmRootmuxGptOsc24m      = 0U, /*!< GPT Clock from OSC 24M.*/
    ccmRootmuxGptEnetPll100m = 1U, /*!< GPT Clock from Ethernet PLL 100M.*/
    ccmRootmuxGptSysPllPfd0  = 2U, /*!< GPT Clock from SYSTEM PLL PFD0.*/
    ccmRootmuxGptEnetPll40m  = 3U, /*!< GPT Clock from Ethernet PLL 40M.*/
    ccmRootmuxGptVideoPll    = 4U, /*!< GPT Clock from VIDEO PLL.*/
    ccmRootmuxGptRef1m       = 5U, /*!< GPT Clock from Refernece Clock 1M.*/
    ccmRootmuxGptAudioPll    = 6U, /*!< GPT Clock from AUDIO PLL.*/
    ccmRootmuxGptExtClk      = 7U, /*!< GPT Clock from External Clock.*/
};

/*! @brief Clock source enumeration for WDOG peripheral. */
enum _ccm_rootmux_wdog
{
```

```
    ccmRootmuxWdogOsc24m         = 0U, /*!< WDOG Clock from OSC 24M.*/
    ccmRootmuxWdogSysPllPfd2Div2 = 1U, /*!< WDOG Clock from SYSTEM PLL PFD2 divided by 2.*/
    ccmRootmuxWdogSysPllDiv4     = 2U, /*!< WDOG Clock from SYSTEM PLL divided by 4.*/
    ccmRootmuxWdogDdrPllDiv2     = 3U, /*!< WDOG Clock from DDR PLL divided by 2.*/
    ccmRootmuxWdogEnetPll125m    = 4U, /*!< WDOG Clock from Ethernet PLL 125M.*/
    ccmRootmuxWdogUsbPll         = 5U, /*!< WDOG Clock from USB PLL.*/
    ccmRootmuxWdogRef1m          = 6U, /*!< WDOG Clock from Refernece Clock 1M.*/
    ccmRootmuxWdogSysPllPfd1Div2 = 7U, /*!< WDOG Clock from SYSTEM PLL PFD1 divided by 2.*/
};

/*! @brief CCM PLL gate control. */
enum _ccm_pll_gate
{
    ccmPllGateCkil      = (uint32_t)(&CCM_PLL_CTRL0),  /*!< Ckil PLL Gate.*/
    ccmPllGateArm       = (uint32_t)(&CCM_PLL_CTRL1),  /*!< ARM PLL Gate.*/
    ccmPllGateArmDiv1   = (uint32_t)(&CCM_PLL_CTRL2),  /*!< ARM PLL Div1 Gate.*/
    ccmPllGateDdr       = (uint32_t)(&CCM_PLL_CTRL3),  /*!< DDR PLL Gate.*/
    ccmPllGateDdrDiv1   = (uint32_t)(&CCM_PLL_CTRL4),  /*!< DDR PLL Div1 Gate.*/
    ccmPllGateDdrDiv2   = (uint32_t)(&CCM_PLL_CTRL5),  /*!< DDR PLL Div2 Gate.*/
    ccmPllGateSys       = (uint32_t)(&CCM_PLL_CTRL6),  /*!< SYSTEM PLL Gate.*/
    ccmPllGateSysDiv1   = (uint32_t)(&CCM_PLL_CTRL7),  /*!< SYSTEM PLL Div1 Gate.*/
    ccmPllGateSysDiv2   = (uint32_t)(&CCM_PLL_CTRL8),  /*!< SYSTEM PLL Div2 Gate.*/
    ccmPllGateSysDiv4   = (uint32_t)(&CCM_PLL_CTRL9),  /*!< SYSTEM PLL Div4 Gate.*/
    ccmPllGatePfd0      = (uint32_t)(&CCM_PLL_CTRL10), /*!< PFD0 Gate.*/
    ccmPllGatePfd0Div2  = (uint32_t)(&CCM_PLL_CTRL11), /*!< PFD0 Div2 Gate.*/
    ccmPllGatePfd1      = (uint32_t)(&CCM_PLL_CTRL12), /*!< PFD1 Gate.*/
    ccmPllGatePfd1Div2  = (uint32_t)(&CCM_PLL_CTRL13), /*!< PFD1 Div2 Gate.*/
    ccmPllGatePfd2      = (uint32_t)(&CCM_PLL_CTRL14), /*!< PFD2 Gate.*/
    ccmPllGatePfd2Div2  = (uint32_t)(&CCM_PLL_CTRL15), /*!< PDF2 Div2.*/
    ccmPllGatePfd3      = (uint32_t)(&CCM_PLL_CTRL16), /*!< PDF3 Gate.*/
    ccmPllGatePfd4      = (uint32_t)(&CCM_PLL_CTRL17), /*!< PDF4 Gate.*/
    ccmPllGatePfd5      = (uint32_t)(&CCM_PLL_CTRL18), /*!< PDF5 Gate.*/
    ccmPllGatePfd6      = (uint32_t)(&CCM_PLL_CTRL19), /*!< PDF6 Gate.*/
    ccmPllGatePfd7      = (uint32_t)(&CCM_PLL_CTRL20), /*!< PDF7 Gate.*/
    ccmPllGateEnet      = (uint32_t)(&CCM_PLL_CTRL21), /*!< Ethernet PLL Gate.*/
    ccmPllGateEnet500m  = (uint32_t)(&CCM_PLL_CTRL22), /*!< Ethernet 500M PLL Gate.*/
    ccmPllGateEnet250m  = (uint32_t)(&CCM_PLL_CTRL23), /*!< Ethernet 250M PLL Gate.*/
    ccmPllGateEnet125m  = (uint32_t)(&CCM_PLL_CTRL24), /*!< Ethernet 125M PLL Gate.*/
    ccmPllGateEnet100m  = (uint32_t)(&CCM_PLL_CTRL25), /*!< Ethernet 100M PLL Gate.*/
    ccmPllGateEnet50m   = (uint32_t)(&CCM_PLL_CTRL26), /*!< Ethernet 50M PLL Gate.*/
    ccmPllGateEnet40m   = (uint32_t)(&CCM_PLL_CTRL27), /*!< Ethernet 40M PLL Gate.*/
    ccmPllGateEnet25m   = (uint32_t)(&CCM_PLL_CTRL28), /*!< Ethernet 25M PLL Gate.*/
    ccmPllGateAudio     = (uint32_t)(&CCM_PLL_CTRL29), /*!< AUDIO PLL Gate.*/
    ccmPllGateAudioDiv1 = (uint32_t)(&CCM_PLL_CTRL30), /*!< AUDIO PLL Div1 Gate.*/
    ccmPllGateVideo     = (uint32_t)(&CCM_PLL_CTRL31), /*!< VIDEO PLL Gate.*/
    ccmPllGateVideoDiv1 = (uint32_t)(&CCM_PLL_CTRL32), /*!< VIDEO PLL Div1 Gate.*/
};

/*! @brief CCM CCGR gate control. */
```

```
enum _ccm_ccgr_gate
{
    ccmCcgrGateSimWakeup = (uint32_t)(&CCM_CCGR9),   /*!< Wakeup Mix Bus Clock Gate.*/
    ccmCcgrGateIpmux1    = (uint32_t)(&CCM_CCGR10),  /*!< IOMUX1 Clock Gate.*/
    ccmCcgrGateIpmux2    = (uint32_t)(&CCM_CCGR11),  /*!< IOMUX2 Clock Gate.*/
    ccmCcgrGateIpmux3    = (uint32_t)(&CCM_CCGR12),  /*!< IPMUX3 Clock Gate.*/
    ccmCcgrGateOcram     = (uint32_t)(&CCM_CCGR17),  /*!< OCRAM Clock Gate.*/
    ccmCcgrGateOcramS    = (uint32_t)(&CCM_CCGR18),  /*!< OCRAM S Clock Gate.*/
    ccmCcgrGateQspi      = (uint32_t)(&CCM_CCGR21),  /*!< QSPI Clock Gate.*/
    ccmCcgrGateAdc       = (uint32_t)(&CCM_CCGR32),  /*!< ADC Clock Gate.*/
    ccmCcgrGateRdc       = (uint32_t)(&CCM_CCGR38),  /*!< RDC Clock Gate.*/
    ccmCcgrGateMu        = (uint32_t)(&CCM_CCGR39),  /*!< MU Clock Gate.*/
    ccmCcgrGateSemaHs    = (uint32_t)(&CCM_CCGR40),  /*!< SEMA HS Clock Gate.*/
    ccmCcgrGateSema1     = (uint32_t)(&CCM_CCGR64),  /*!< SEMA1 Clock Gate.*/
    ccmCcgrGateSema2     = (uint32_t)(&CCM_CCGR65),  /*!< SEMA2 Clock Gate.*/
    ccmCcgrGateCan1      = (uint32_t)(&CCM_CCGR116), /*!< CAN1 Clock Gate.*/
    ccmCcgrGateCan2      = (uint32_t)(&CCM_CCGR117), /*!< CAN2 Clock Gate.*/
    ccmCcgrGateEcspi1    = (uint32_t)(&CCM_CCGR120), /*!< ECSPI1 Clock Gate.*/
    ccmCcgrGateEcspi2    = (uint32_t)(&CCM_CCGR121), /*!< ECSPI2 Clock Gate.*/
    ccmCcgrGateEcspi3    = (uint32_t)(&CCM_CCGR122), /*!< ECSPI3 Clock Gate.*/
    ccmCcgrGateEcspi4    = (uint32_t)(&CCM_CCGR123), /*!< ECSPI4 Clock Gate.*/
    ccmCcgrGateGpt1      = (uint32_t)(&CCM_CCGR124), /*!< GPT1 Clock Gate.*/
    ccmCcgrGateGpt2      = (uint32_t)(&CCM_CCGR125), /*!< GPT2 Clock Gate.*/
    ccmCcgrGateGpt3      = (uint32_t)(&CCM_CCGR126), /*!< GPT3 Clock Gate.*/
    ccmCcgrGateGpt4      = (uint32_t)(&CCM_CCGR127), /*!< GPT4 Clock Gate.*/
    ccmCcgrGateI2c1      = (uint32_t)(&CCM_CCGR136), /*!< I2C1 Clock Gate.*/
    ccmCcgrGateI2c2      = (uint32_t)(&CCM_CCGR137), /*!< I2C2 Clock Gate.*/
    ccmCcgrGateI2c3      = (uint32_t)(&CCM_CCGR138), /*!< I2C3 Clock Gate.*/
    ccmCcgrGateI2c4      = (uint32_t)(&CCM_CCGR139), /*!< I2C4 Clock Gate.*/
    ccmCcgrGateUart1     = (uint32_t)(&CCM_CCGR148), /*!< UART1 Clock Gate.*/
    ccmCcgrGateUart2     = (uint32_t)(&CCM_CCGR149), /*!< UART2 Clock Gate.*/
    ccmCcgrGateUart3     = (uint32_t)(&CCM_CCGR150), /*!< UART3 Clock Gate.*/
    ccmCcgrGateUart4     = (uint32_t)(&CCM_CCGR151), /*!< UART4 Clock Gate.*/
    ccmCcgrGateUart5     = (uint32_t)(&CCM_CCGR152), /*!< UART5 Clock Gate.*/
    ccmCcgrGateUart6     = (uint32_t)(&CCM_CCGR153), /*!< UART6 Clock Gate.*/
    ccmCcgrGateUart7     = (uint32_t)(&CCM_CCGR154), /*!< UART7 Clock Gate.*/
    ccmCcgrGateWdog1     = (uint32_t)(&CCM_CCGR156), /*!< WDOG1 Clock Gate.*/
    ccmCcgrGateWdog2     = (uint32_t)(&CCM_CCGR157), /*!< WDOG2 Clock Gate.*/
    ccmCcgrGateWdog3     = (uint32_t)(&CCM_CCGR158), /*!< WDOG3 Clock Gate.*/
    ccmCcgrGateWdog4     = (uint32_t)(&CCM_CCGR159), /*!< WDOG4 Clock Gate.*/
    ccmCcgrGateGpio1     = (uint32_t)(&CCM_CCGR160), /*!< GPIO1 Clock Gate.*/
    ccmCcgrGateGpio2     = (uint32_t)(&CCM_CCGR161), /*!< GPIO2 Clock Gate.*/
    ccmCcgrGateGpio3     = (uint32_t)(&CCM_CCGR162), /*!< GPIO3 Clock Gate.*/
    ccmCcgrGateGpio4     = (uint32_t)(&CCM_CCGR163), /*!< GPIO4 Clock Gate.*/
    ccmCcgrGateGpio5     = (uint32_t)(&CCM_CCGR164), /*!< GPIO5 Clock Gate.*/
    ccmCcgrGateGpio6     = (uint32_t)(&CCM_CCGR165), /*!< GPIO6 Clock Gate.*/
    ccmCcgrGateGpio7     = (uint32_t)(&CCM_CCGR166), /*!< GPIO7 Clock Gate.*/
    ccmCcgrGateIomux     = (uint32_t)(&CCM_CCGR168), /*!< IOMUX Clock Gate.*/
    ccmCcgrGateIomuxLpsr = (uint32_t)(&CCM_CCGR169), /*!< IOMUX LPSR Clock Gate.*/
```

```
};

/*! @brief CCM gate control value. */
enum _ccm_gate_value
{
    ccmClockNotNeeded     = 0x0U,    /*!< Clock always disabled.*/
    ccmClockNeededRun     = 0x1111U, /*!< Clock enabled when CPU is running.*/
    ccmClockNeededRunWait = 0x2222U, /*!< Clock enabled when CPU is running or in WAIT mode.*/
    ccmClockNeededAll     = 0x3333U, /*!< Clock always enabled.*/
};

/*******************************************************************************
 * API
 ******************************************************************************/

#if defined(__cplusplus)
extern "C" {
#endif

/*!
 * @name CCM Root Setting
 * @{
 */

/*!
 * @brief Set clock root mux
 *
 * @param base CCM base pointer.
 * @param ccmRoot Root control (see @ref _ccm_root_control enumeration)
 * @param mux Root mux value (see @ref _ccm_rootmux_xxx enumeration)
 */
static inline void CCM_SetRootMux(CCM_Type * base, uint32_t ccmRoot, uint32_t mux)
{
    CCM_REG(ccmRoot) = (CCM_REG(ccmRoot) & (~CCM_TARGET_ROOT_MUX_MASK)) |
                               CCM_TARGET_ROOT_MUX(mux);
}

/*!
 * @brief Get clock root mux
 *
 * @param base CCM base pointer.
 * @param ccmRoot Root control (see @ref _ccm_root_control enumeration)
 * @return root mux value (see @ref _ccm_rootmux_xxx enumeration)
 */
static inline uint32_t CCM_GetRootMux(CCM_Type * base, uint32_t ccmRoot)
{
    return (CCM_REG(ccmRoot) & CCM_TARGET_ROOT_MUX_MASK) >> CCM_TARGET_ROOT_MUX_SHIFT;
}

```

```
373 /*!
374  * @brief Enable clock root
375  *
376  * @param base CCM base pointer.
377  * @param ccmRoot Root control (see @ref _ccm_root_control enumeration)
378  */
379 static inline void CCM_EnableRoot(CCM_Type * base, uint32_t ccmRoot)
380 {
381     CCM_REG_SET(ccmRoot) = CCM_TARGET_ROOT_SET_ENABLE_MASK;
382 }
383
384 /*!
385  * @brief Disable clock root
386  *
387  * @param base CCM base pointer.
388  * @param ccmRoot Root control (see @ref _ccm_root_control enumeration)
389  */
390 static inline void CCM_DisableRoot(CCM_Type * base, uint32_t ccmRoot)
391 {
392     CCM_REG_CLR(ccmRoot) = CCM_TARGET_ROOT_CLR_ENABLE_MASK;
393 }
394
395 /*!
396  * @brief Check whether clock root is enabled
397  *
398  * @param base CCM base pointer.
399  * @param ccmRoot Root control (see @ref _ccm_root_control enumeration)
400  * @return CCM root enabled or not.
401  *         - true: Clock root is enabled.
402  *         - false: Clock root is disabled.
403  */
404 static inline bool CCM_IsRootEnabled(CCM_Type * base, uint32_t ccmRoot)
405 {
406     return (bool)(CCM_REG(ccmRoot) & CCM_TARGET_ROOT_ENABLE_MASK);
407 }
408
409 /*!
410  * @brief Set root clock divider
411  *
412  * @param base CCM base pointer.
413  * @param ccmRoot Root control (see @ref _ccm_root_control enumeration)
414  * @param pre Pre divider value (0-7, divider=n+1)
415  * @param post Post divider value (0-63, divider=n+1)
416  */
417 void CCM_SetRootDivider(CCM_Type * base, uint32_t ccmRoot, uint32_t pre, uint32_t post);
418
419 /*!
420  * @brief Get root clock divider
421  *
```

```
 * @param base CCM base pointer.
 * @param ccmRoot Root control (see @ref _ccm_root_control enumeration)
 * @param pre Pointer to pre divider value store address
 * @param post Pointer to post divider value store address
 */
void CCM_GetRootDivider(CCM_Type * base, uint32_t ccmRoot, uint32_t *pre, uint32_t *post);

/*!
 * @brief Update clock root in one step, for dynamical clock switching
 *
 * @param base CCM base pointer.
 * @param ccmRoot Root control (see @ref _ccm_root_control enumeration)
 * @param root mux value (see @ref _ccm_rootmux_xxx enumeration)
 * @param pre Pre divider value (0-7, divider=n+1)
 * @param post Post divider value (0-63, divider=n+1)
 */
void CCM_UpdateRoot(CCM_Type * base, uint32_t ccmRoot, uint32_t mux, uint32_t pre, uint32_t post);

/*@}*/

/*!
 * @name CCM Gate Control
 * @{
 */

/*!
 * @brief Set PLL or CCGR gate control
 *
 * @param base CCM base pointer.
 * @param ccmGate Gate control (see @ref _ccm_pll_gate and @ref _ccm_ccgr_gate enumeration)
 * @param control Gate control value (see @ref _ccm_gate_value)
 */
static inline void CCM_ControlGate(CCM_Type * base, uint32_t ccmGate, uint32_t control)
{
    CCM_REG(ccmGate) = control;
}

/*@}*/

#if defined(__cplusplus)
}
#endif

/*! @}*/

#endif /* __CCM_IMX7D_H__ */
/*******************************************************************************
 * EOF
 ******************************************************************************/
```